Most microprocessors currently have only one stack. This is used mainly for the implementation of subroutines. The use of the stack has been expanded to pass parameters between subroutines or for temporary data storage as an alternative to RAM storage. Using the stack for this purpose has two disadvantages. First, to get at the parameters once the routine has been called, requires manipulation of the stack. This leads to the second possible disadvantage. If an interrupt were to occur during manipulation of the stack , problems could arise. Use of RAM requires software manipulation instead of simply"pushing" or "popping".

A possible solution to the problem is to provide a separate stack or LIFO (Last in First out) memory configured as a single memory or I/O location. A stack may be realised using an up/down counter to access RAM. A push to the stack will increment the counter afterwriting to the addressed location. A pop decrements the counter and outputs the new addressed location. Figure 1 demonstrates the idea in principle.

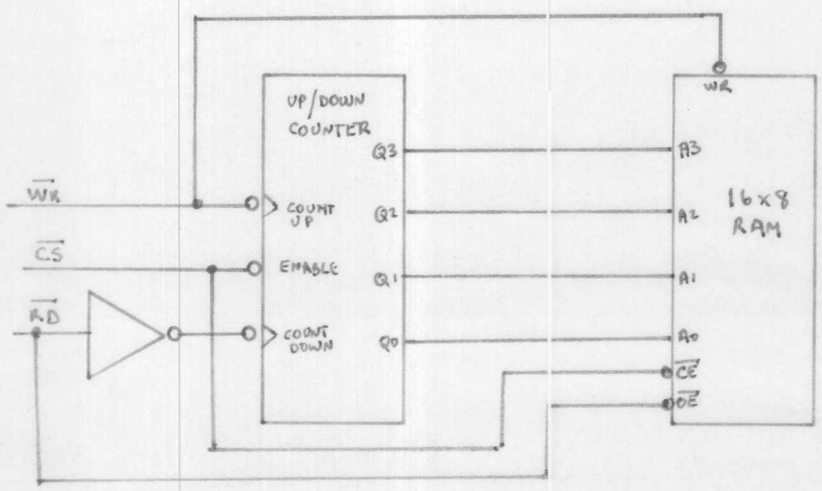

FIGURE 1

A SIMPLE HARDWARE STACK

SBQ SHOULD IN ALL INSTANCES BE REPLACED BY QBX

By setting the counter (accessing the RAM) any byte or area of stack can be accessed. The EEPROM may be written to a finite number of times (1000 minimum).

This also allows bit alterable memory by reading back the EEPROM, changing the bit and writing back to EEPROM.

Since the FOG may be accessed as a single Port D.M.A. techniques may be used.

This product is made for SBX*, STD and S100 buses. The latter two include address or I/O decoding.

* Registered trademark of Intel Corporation

** Conceptual only-does not exist for FOG/C and is on chip with the RAM for the FOG/E

*** For FOG/C

This extra stack becomes independent of interrupts and requires no manipulation of the stack present within the microprocessor.

Quantum Electronic Design has introduced several modules working on this principle, but with several improvements.

The stack size has been increased to 256 bytes and the counter has been made programmable. This allows multiple stacks to be implemented with areas within the stack allocated to different subroutines. Further, it is possible to read back the stack pointer and this may allow information to be conveyed. For instance, if a block of characters was to be output and passed via the stack, when the counter sets to zero the output process is completed.

The addition of this stack does not place a strain on the RAM resources and only occupies several bytes of memory or I/O space.

To this point the modules are identical in principle. Both the modules are non-volatile. The FOG/C is C-MOS RAM that requires battery backup, and requires no further explanation. The FOG/E has a 256 byte RAM with a bit for bit EEPROM that requires only 5 V to programme. The RAM can be treated conventionally, but on the generation of a store pulse the whole RAM is copied into EEPROM. A recall pulse copies the EEPROM back into RAM. (Overwriting any previous data).

This opens the door to many possible aplications. The RAM can be treated normally with the EEPROM used only for loading constants on power up. Data may be transferred at any time and as many times as necessary from the EEPROM to the RAM.

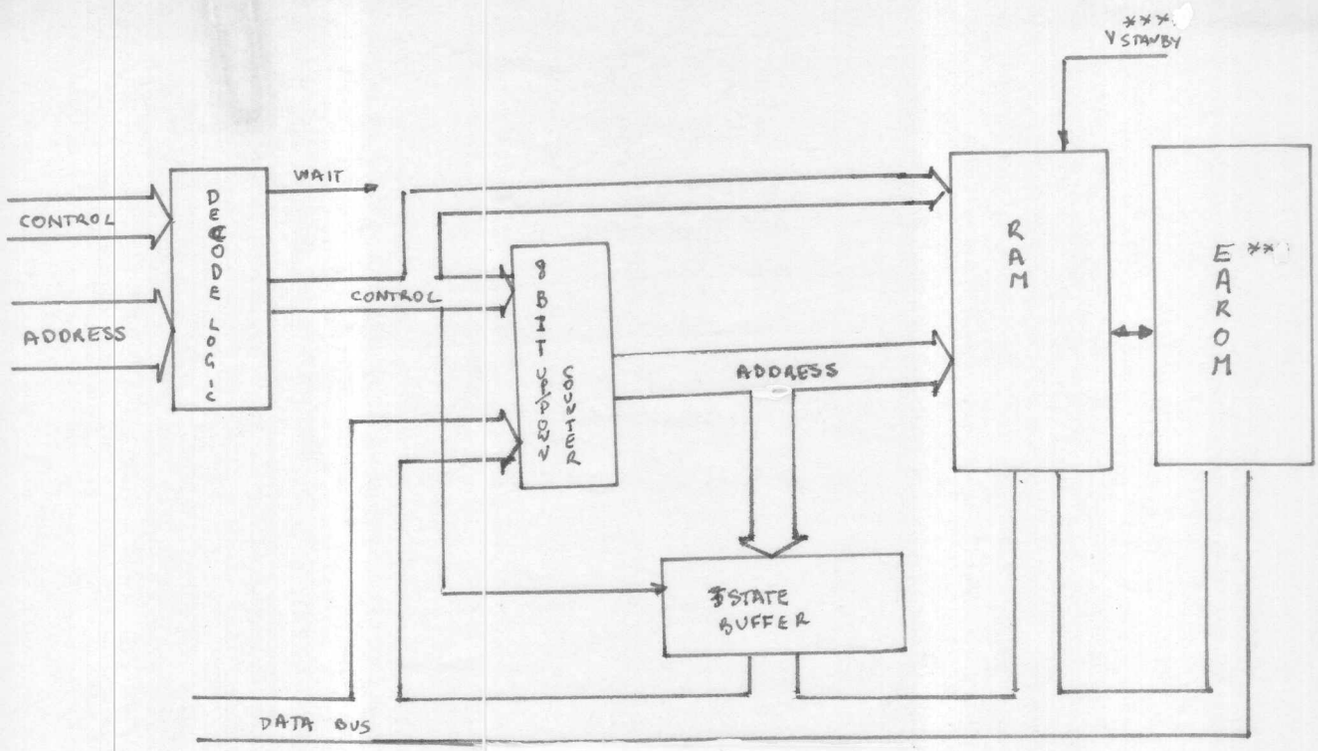

FIGURE 2

BLOCK DIAGRAM FOR SBQ-FOG